Using Gyroscopic Phenomena for Attitude Control
3C5 Full Technical Report

Zach Lambert
Pembroke College
08/12/19

1 Abstract

The objective of this report was to investigate how gyroscopic phenomena can be utilised in spacecrafts for attitude control, as an interesting application of gyroscopes. This included reaction wheels, which use rotors with variable spin, and control moment gyroscopes, which use rotors with variable spin axes. In addition to deriving equations of motion that describe these systems and finding solutions, numerical integration was used to validate these and simulate less predictable behaviour. It was found that reaction wheels can control angular velocity along the same axis as the rotor by varying the rotor spin, although the solution become more complex when using 3 reaction wheels. On the other hand, control moment gyroscopes controlled angular velocity along an axis perpendicular to the spin axis, but can't sustain stable rotation for prolonged periods of time.

2 Introduction

This report investigates how gyroscopic phenomena can be used for attitude control in spacecrafts. The two actuators of interest are reaction wheels and control moment gyroscopes (CMGs), which are commonly used [1] [2].

Both reaction wheels and CMGs exploit conservation of angular momentum to control the angular velocity of the spacecraft. By changing the angular momentum of gyroscopes within the spacecraft, the angular momentum of the whole spacecraft must experience an equal and opposite change, such that the total angular momentum remains constant [3].

Reaction wheels use spinning rotors with variable angular velocity. They change the angular velocity of the gyroscopes by changing the rotor spin, while the spin axis remains fixed relative to the spacecraft.

Control moment gyroscopes utilise gyroscopic precession to rotate the spacecraft. They use rotors mounted in single gimbals, allowing the spin axis to be controlled about one axis, while the rotor operates at fixed spin. By tilting the spin axis of the rotor, this changes the angular momentum of the gyroscope.

The advantage of gyroscopic devices for attitude control over thrusters, is that they are driven by motors, which use electricity that can be generated by solar panels. On the other hand, thrusters use propellent which cannot be recovered.

3 Method

3.1 Finding Analytical Solutions

For each system investigated, the first step was to find the equations of motion. Following this, analytical solutions could be found for a variety of types of behaviour.

Additionally, by adding perturbations to the equilbrium solutions, the stability of solutions could be investigated.

3.2 Validating Solutions with Numerical Integration

In order to validate that the solutions found are sensible, as well as investigate more complex motion, numerical integration was used to integrate the equations of motion.

Since the equations of motion are all first order, a single application of the fourth-order Runge-Kutta formula (RK4) can be used, described below [4].
If the equations of motion can be described by $\dot{\boldsymbol{y}} = f(t, \boldsymbol{y})$, then $\boldsymbol{y}$ is integrated using this formula :

$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{y}(t_n) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\Delta t + O(\Delta t^5)$$

Where:

$$\begin{aligned} k_1 &= f(t_n, \boldsymbol{y}(t_n)) \\ k_2 &= f(t_n + \Delta t/2, \boldsymbol{y}(t_n) + k_1 \Delta t/2) \\ k_3 &= f(t_n + \Delta t/2, \boldsymbol{y}(t_n) + k_2 \Delta t/2) \\ k_4 &= f(t_n + \Delta t, \boldsymbol{y}(t_n) + k_3 \Delta t) \end{aligned}$$

To check a given solution, the integrator needs to be supplied the function f and the initial conditions.

The function f includes time dependence because of properties of the system which can vary in time. For the reaction wheels, the rotor spins could vary, and for the control moment gyroscopes, the spin axis inclination could vary.

This was implemented in Python, the details of which can be seen in the appendix.

4 Results

4.1 Motion of the Spacecraft with No Actuators

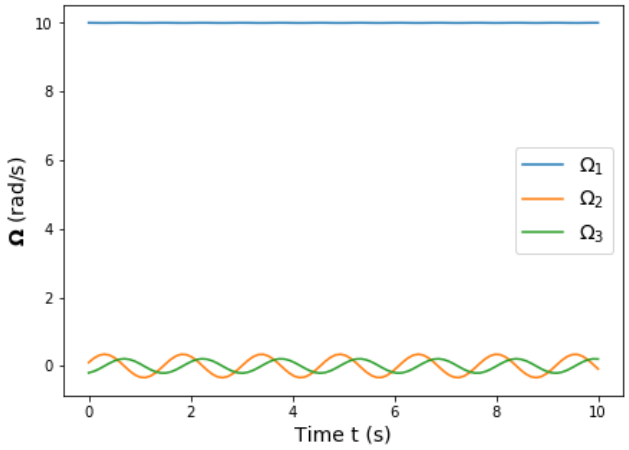

Figure 1: Stable motion of an ABC body with constant angular velocity about a principle axis. Stable motion occurs when this principle axis has the minimum or maximum moment of inertia

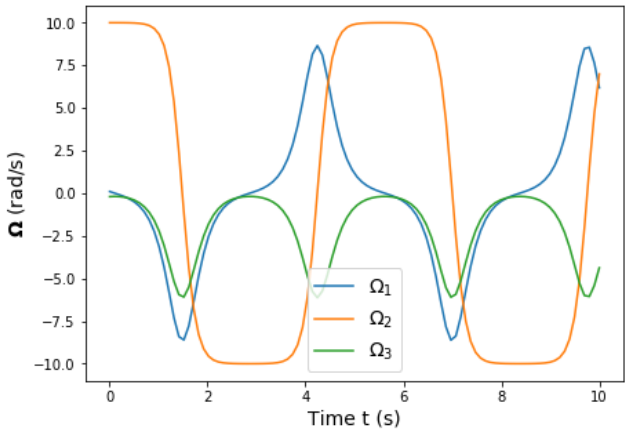

Figure 2: Unstable motion of an ABC body. Unstable motion occurs when the body attempts to rotate about the principle axis with intermediate moment of inertia.

4.2 Motion of the Spacecraft with Reaction Wheels

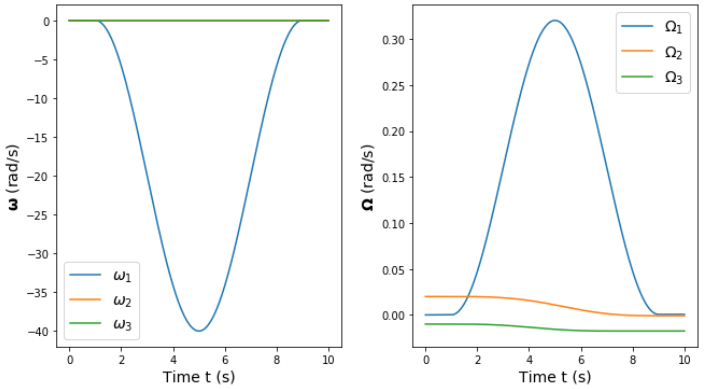

Figure 3: Example of stable rotation about an axis using a single reaction wheel with variable spin. Perturbations in the other axes follow stable oscillation.

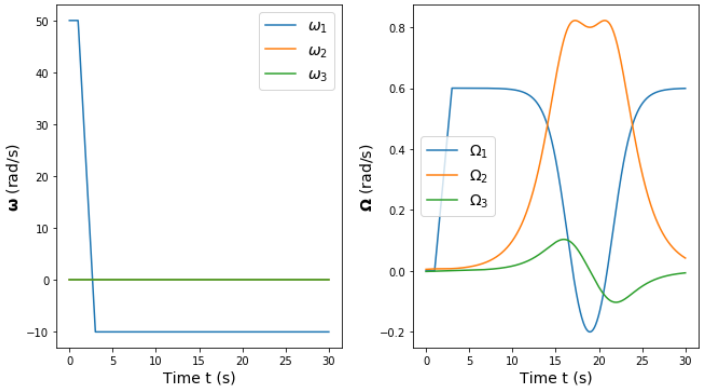

Figure 4: Example of unstable rotation about a axis using a single reaction wheel with variable spin. Perturbations in the other axes grow exponentially and cause the motion to become unpredictable.

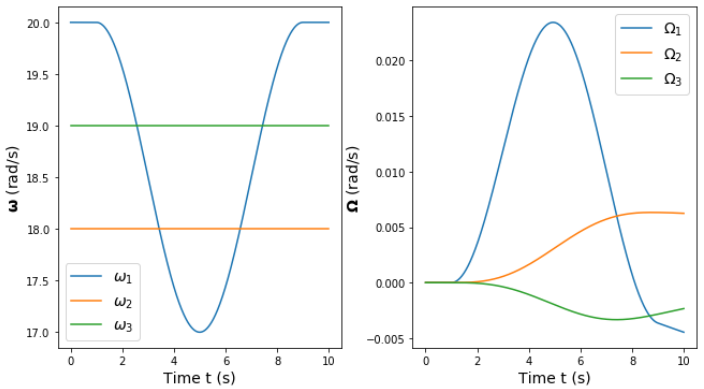

Figure 5: Demonstrates what happens if there is attempted rotation about a axis using a single reaction wheel when the other reaction wheels are spinning too. In this case, the spin of the other reaction wheels need to be changed too.

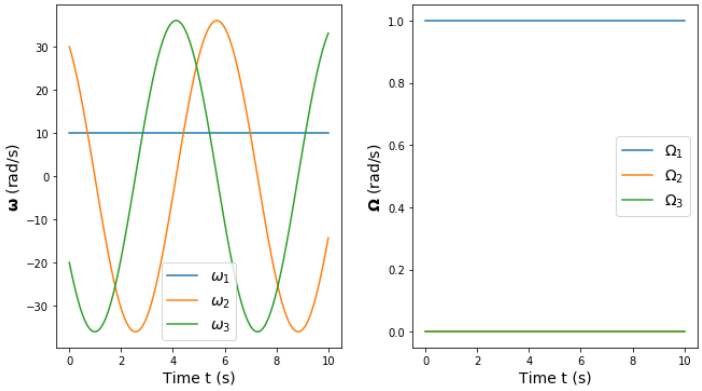

Figure 6: Demonstrates the correct method of rotating about a single axis using reaction wheels when all the reaction wheels have spin. In this case, the spacecraft has constant angular velocity about an axis.

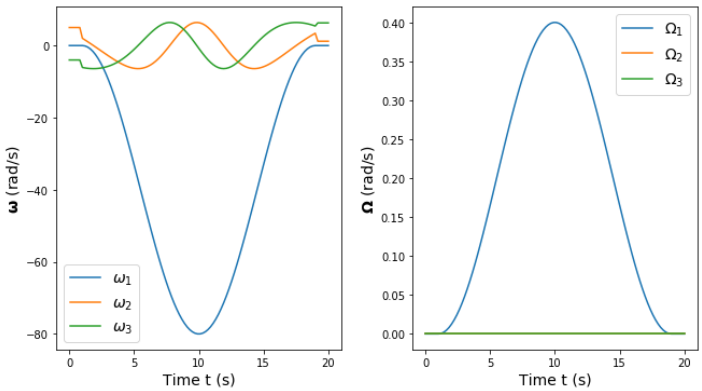

Figure 7: Shows how reaction wheels can be used to rotate the spacecraft about a single axis with a varying angular velocity.

4.3 Motion of the Spacecraft with Control Moment Gyroscopes

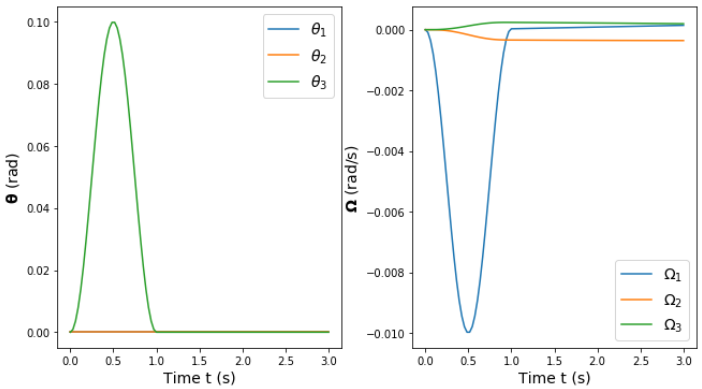

Figure 8: Demonstrates how a set of control moment gyroscopes can be used to provide controlled rotation about an axis. In this case, the changes in angular velocity about the other two axes have been kept small, so don't affect the motion significantly.

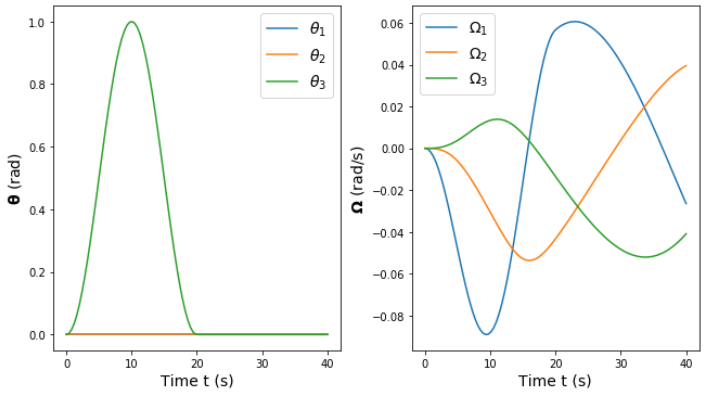

Figure 9: Demonstrates how the the control moment gyroscopes can produce uncontrolled motion. In this case, the angular velocity about other axes was allowed to become too large, which caused the motion to become uncontrolled.

5 Discussion

5.1 Motion of the Spacecraft with No Actuators

5.1.1 Equations of Motion

Define I as the inertia matrix of the spacecraft about the centre of mass and $\boldsymbol{\Omega}$ as its angular velocity. This means that the angular momentum about the centre of mass is given by [5]:

$$\boldsymbol{h} = I\boldsymbol{\Omega} \tag{1}$$

In order for this inertia matrix to be diagonal, therefore simplifying the analysis, vectors must be defined relative to the principle axis of the spacecraft, defined as $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$.

To find the equations of motions, we used the equation [6]:

$$\boldsymbol{Q}_P = \dot{\boldsymbol{h}}_P + \boldsymbol{r}_P \times \boldsymbol{p} \tag{2}$$

However, since angular momentum is taken about the centre of mass ($\boldsymbol{r}_P \times \boldsymbol{p} = 0$), this simplifies to:

$$\boldsymbol{Q} = \dot{\boldsymbol{h}} \tag{3}$$

This is only valid if $\dot{\boldsymbol{h}}$ is defined within a fixed reference frame. However the expression for $\boldsymbol{h}$ given in eq. (1) is defined in the reference frame of the spacecraft, which is rotating with angular velocity Ω. To evaluate $\dot{\boldsymbol{h}}$ in a fixed reference frame F, when defined in a rotating reference frame R, the following is used [7]:

$$\left[\dot{\boldsymbol{h}}\right]_F = \left[\dot{\boldsymbol{h}}\right]_R + \boldsymbol{\Omega} \times [\boldsymbol{h}]_R \tag{4}$$

Combining eq. (1), eq. (2) and eq. (4) gives:

$$\boldsymbol{Q} = \dot{\boldsymbol{h}} = I\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega} \tag{5}$$

Expanding this out gives the equations of motion for the spacecraft, which are Euler's equations:

$$\begin{aligned} Q_1 &= A\dot{\Omega}_1 + (C-B)\Omega_2\Omega_3 \\ Q_2 &= B\dot{\Omega}_2 + (A-C)\Omega_1\Omega_3 \\ Q_3 &= C\dot{\Omega}_3 + (B-A)\Omega_1\Omega_2 \end{aligned} \tag{6}$$

Since all the systems being investigated have no external torques, the $\boldsymbol{Q}$ will always be zero.

5.1.2 Constant Angular Velocity Solution

In order for the spacecraft to rotate about its axes, it will need to be able to rotate at constant angular velocity about its axes. From eq. (6) it can be seen that having constant angular velocity about a single axis is a valid solution. For example, set $\boldsymbol{\Omega} = \Omega\boldsymbol{i}$ = Constant and $\boldsymbol{Q} = 0$, then substitute into eq. (6):

$$\begin{aligned} 0 &= 0 + (C-B) \times 0 \times 0 = 0 \\ 0 &= 0 + (A-C) \times \Omega_1 \times 0 = 0 \\ 0 &= 0 + (B-A) \times \Omega_1 \times 0 = 0 \end{aligned}$$

Therefore, in the absence of external couples, there can be constant motion about a single principle axis. However, only certain solutions are stable. To investigate stability, set Ω_2 and Ω_3 to have small perturbations:

$$\boldsymbol{\Omega} = \begin{pmatrix} \Omega_1 \\ \delta\Omega_2 \\ \delta\Omega_3 \end{pmatrix} \tag{7}$$

Substituting eq. (7) into eq. (6) and ignoring second order terms of perturations gives:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 \\ 0 &= B\delta\dot{\Omega}_2 + (A-C)\Omega_1\delta\Omega_3 \\ 0 &= C\delta\dot{\Omega}_3 + (B-A)\Omega_1\delta\Omega_2 \end{aligned} \tag{8}$$

From eq. (8) we get a pair of coupled first order equations for $\delta\Omega_2$ and $\delta\Omega_3$:

$$\begin{aligned} \delta\dot{\Omega}_2 &= \Omega_1\frac{C-A}{B}\delta\Omega_3 \\ \delta\dot{\Omega}_3 &= -\Omega_1\frac{B-A}{C}\delta\Omega_2 \end{aligned} \tag{9}$$

These can be combined to give second order differential equations for $\delta\Omega_2$ and $\delta\Omega_3$.

$$\begin{aligned} \delta\ddot{\Omega}_2 &= -\Omega_1^2\frac{(B-A)(C-A)}{BC}\delta\Omega_2 \\ \delta\ddot{\Omega}_3 &= -\Omega_1^2\frac{(B-A)(C-A)}{BC}\delta\Omega_3 \end{aligned} \tag{10}$$

The solutions to these equations are:

$$\delta\Omega_2 = k\sin(\omega t + \phi) \qquad \delta\Omega_3 = k\cos(\omega t + \phi) \tag{11}$$

Where A and ϕ depend on initial conditions, and:

$$\omega^2 = \Omega_1^2\frac{(B-A)(C-A)}{BC} \tag{12}$$

In order for this to be a valid solution and therefore be stable, ω^2 must be positive. Otherwise, $\delta\Omega_2$ and $\delta\Omega_3$ will grow exponentially and be unstable. In the case that $A < B < C$, ω^2 will be positive when rotating about the $\boldsymbol{i}$ or $\boldsymbol{k}$ axis, but negative for the $\boldsymbol{j}$ axis. Therefore, the body can have stable rotation about its principle axis with the minimum or maximum moment of inertia, but not the intermediate moment of inertia.

For constant angular velocity of an ABC body about a principle axis, Figure 1 shows the results of a simulation about a stable axis and Figure 2 shows the results of a simulation about an unstable axis, when the other components of angular velocity have been given small perturbations.

5.2 Motion of the Spacecraft with Reaction Wheels

5.2.1 Structure of the System

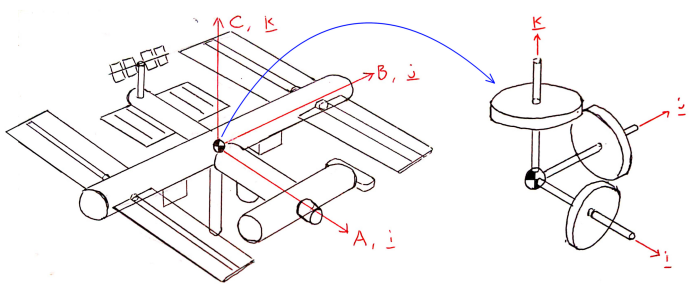

Figure 10: Diagram of the space station and attached gyroscopes with principle axis and principle moments of inertia labelled.

This modifies the system looked at previously by adding 3 rotors at the centre of mass, aligned with the principle axes. Each gyroscope is an AAC body, with identical moments of inertia A_R and C_R perpendicular to, and along the spin axes. Their inertia matrices are I_{R1}, I_{R2} and I_{R3}, with the same AAC form, but C_R in different positions.

These moments of inertia are releative to the centre of mass of the spacecraft. These can be related to the moments of inertia about the centre of mass of the rotors $A_{R,G}$ and $C_{R,G}$ by the parallel axis theorem [8]:

$$C_R = C_{R,G} \qquad A_R = A_{R,G} + d^2 m \tag{13}$$

Where d is the distance along the spin axis from the centre of mass of the rotor to the centre of mass of the spacecraft, and m is the mass of a rotor.

The rotors have spins ω_1, ω_2 and ω_3.

5.2.2 Equations of Motion

If the whole spacecraft is rotating with angular velocity Ω and the reaction wheels are fixed to the spacecraft, then they will be rotating with angular velocities:

$$\begin{aligned} \text{Angular velocity of rotor 1} &= \boldsymbol{\Omega} + \omega_1 \boldsymbol{i} \\ \text{Angular velocity of rotor 2} &= \boldsymbol{\Omega} + \omega_2 \boldsymbol{j} \\ \text{Angular velocity of rotor 3} &= \boldsymbol{\Omega} + \omega_3 \boldsymbol{k} \end{aligned} \tag{14}$$

Therefore, the total angular momentum, including the spacecraft and rotors can be given by:

$$\begin{aligned} \boldsymbol{h} &= I\boldsymbol{\Omega} + I_{R1}(\boldsymbol{\Omega} + \omega_1 \boldsymbol{i}) + I_{R2}(\boldsymbol{\Omega} + \omega_2 \boldsymbol{j}) + I_{R3}(\boldsymbol{\Omega} + \omega_3 \boldsymbol{k}) \\ \boldsymbol{h} &= (I + I_{R1} + I_{R2} + I_{R3})\boldsymbol{\Omega} + C_R(\omega_1 \boldsymbol{i} + \omega_2 \boldsymbol{j} + \omega_3 \boldsymbol{k}) \end{aligned}$$

If I is refefined to include the rotor inertia too, then this can be simplified to:

$$\boldsymbol{h} = I\boldsymbol{\Omega} + C_R\boldsymbol{\omega} \tag{15}$$

Where $\boldsymbol{\omega}$ is a vector containing the gyroscope spins.

From this, the equations of motion are derived in the same way as before, by using eq. (3) ($\dot{\boldsymbol{h}} = 0$) and differentiating $\boldsymbol{h}$ in a rotating reference frame using eq. (4). This gives:

$$0 = I\dot{\boldsymbol{\Omega}} + C_R\dot{\boldsymbol{\omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega} + C\boldsymbol{\Omega} \times \boldsymbol{\omega} \tag{16}$$

Expanding this out gives the full equations of motion for the reaction wheel system:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + C_R\dot{\omega}_1 + (C - B)\Omega_2\Omega_3 + C_R(\Omega_2\omega_3 - \Omega_3\omega_2) \\ 0 &= B\dot{\Omega}_2 + C_R\dot{\omega}_2 + (A - C)\Omega_1\Omega_3 + C_R(\Omega_3\omega_1 - \Omega_1\omega_3) \\ 0 &= C\dot{\Omega}_3 + C_R\dot{\omega}_3 + (B - A)\Omega_1\Omega_2 + C_R(\Omega_1\omega_2 - \Omega_2\omega_1) \end{aligned} \tag{17}$$

5.2.3 Simple Solutions

The Spacecraft is Stationary

Setting $\boldsymbol{\Omega} = \boldsymbol{0}$ gives:

$$\begin{aligned} 0 &= C_R\dot{\omega}_1 \\ 0 &= C_R\dot{\omega}_2 \\ 0 &= C_R\dot{\omega}_3 \end{aligned}$$

This is a solution if $\boldsymbol{\omega}$ is constant. Therefore, if the station has no angular velocity and keeps the rotor spins constant, then it will remain stationary.

The Rotors have Zero Spin

If the rotors have no spin ($\boldsymbol{\omega}$=0), then the equations of motion simplify to Euler's equations, shown in eq. (6), but with moments of inertia for the combined system instead. Thefefore, the motion is the same as if there were no reaction wheels, as expected.

5.2.4 Controlling Rotation using a Single Reaction Wheel

Solution for Rotation about a Single Axis

Setting $\omega_2 = 0$ and $\omega_3 = 0$:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + C_R\dot{\omega}_1 + (C - B)\Omega_2\Omega_3 \\ 0 &= B\dot{\Omega}_2 + (A - C)\Omega_1\Omega_3 + C_R\Omega_3\omega_1 \\ 0 &= C\dot{\Omega}_3 + (B - A)\Omega_1\Omega_2 - C_R\Omega_2\omega_1 \end{aligned} \tag{18}$$

Similar to the rotation about a single axis with no reaction wheels, one solution is to have $\Omega_1 = \text{Constant}, \Omega_2 = 0, \Omega_3 = 0$, giving:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + C_R\dot{\omega}_1 \\ 0 &= 0 \\ 0 &= 0 \end{aligned} \tag{19}$$

Therefore:

$$\begin{aligned}\dot{\Omega}_1 &= -\frac{C_R}{A}\dot{\omega}_1 \\ \Delta\Omega_1 &= -\frac{C_R}{A}\Delta\omega_1\end{aligned} \tag{20}$$

This means that if ω_1 is changed then there must be a corresponding change of Ω_1 in the other direction, by conservation of angular momentum.

Stability

Like before, perturbations $\delta\Omega_2$ and $\delta\Omega_3$ can be added, to check the stability of the solution:

$$\begin{aligned}0 &= A\dot{\Omega}_1 + C_R\dot{\omega}_1 \\ 0 &= B\dot{\Omega}_2 + (A-C)\Omega_1\dot{\Omega}_3 + C_R\delta\Omega_3\omega_1 \\ 0 &= C\dot{\Omega}_3 + (B-A)\Omega_1\dot{\Omega}_2 - C_R\delta\Omega_3\omega_2\end{aligned} \tag{21}$$

This gives the same solution as eq. (11), except that the frequency is different:

$$\omega^2 = \frac{1}{BC}[C_R\omega_1 - (C-A)\Omega_1][C_R\omega_1 - (B-A)\Omega_1] \tag{22}$$

This expression can be simplified by relating ω_1 and Ω_1 from eq. (20):

$$\begin{aligned}A\dot{\Omega}_1 + C\dot{\omega}_1 &= 0 \\ A(\Omega_1 - \Omega_{1,0}) + C_R(\omega_1 - \omega_{1,0}) &= 0 \\ A\Omega_1 + C_R\omega_1 - h_{1,0} &= 0 \\ \Omega_1 &= -\frac{C_R\omega_1 - h_{1,0}}{A}\end{aligned} \tag{23}$$

Where $h_{1,0}$ is the intial angular momentum of the system about the $\boldsymbol{i}$ axis. For example, this will be non-zero if the rotor is spinning initially.

In the case that $h_{1,0} = 0$, substituting eq. (23) into the expression for ω^2 from eq. (22) gives:

$$\omega^2 = \left(\frac{C_R\omega_1}{A}\right)^2 \tag{24}$$

Therefore, if there is no initial angular momentum, then constant angular velocity about a principle axis is always stable, even about an axis with intermediate moment of inertia.

If $h_{1,0} \neq 0$, then the expression becomes more complicated:

$$\omega^2 = \left(\frac{C_R\omega_1}{A}\right)^2 \left[1 - \left(1 - \frac{A}{C}\right)\frac{h_{1,0}}{C_R\omega_1}\right]\left[1 - \left(1 - \frac{A}{B}\right)\frac{h_{1,0}}{C_R\omega_1}\right] \tag{25}$$

In this case, the condition for stability depends on the relative size of the moments of inertia and the ratio $h_{1,0}/C_R\omega_1$, so there is no simple rule. However, some observations are:

- If the gyroscope operates at large spins such that $C_R\omega_1 >> h_{1,0}$, then this is stable.
- If the space station is an AAA body, then it is always stable.
- If the moments of inertia are all of similar magnitudes and $h_{1,0}/C_R\omega_1$ doesn't becomes large, then this is always stable.

Stability of this solution was investigated with simulated, where Figure 3 gives an example of stable motion and Figure 4 gives an example of unstable motion.

5.3 Controlling Rotation using All Reaction Wheels

Validity of the Solution for a Single Axis Reaction Wheel

If the initial angular momentum of the spacecraft is zero, then the reaction wheels can rotate about any principle axis by ensuring that the rotor spin and angular velocity along other axes are zero.

The problem with this, is that it is unlikely that the spacecraft has no net angular momentum. Any sort of disturbance will apply an external torque and change angular momentum.

Therefore, it is desirable to come up with a method for rotating about a given axis when the net angular momentum is non-zero. To simplify the analysis, we can start at an initial condition of $\boldsymbol{\Omega} = 0$, but $\boldsymbol{\omega} \neq \boldsymbol{0}$ where the reaction wheels have absorbed all the angular momentum.

It can be seen from eq. (17) that if all the reaction wheel spins are non-zero then the motion becomes unpredictable if attempting to rotate about a single axis by changing a single rotor spin.. Figure 5 shows an example of what happens if attempting to increase Ω_1 by decreasing ω_1 when the other reaction wheels are spinning too.

Therefore, a solution for changing Ω_1 must involve chaning ω_2 and ω_3 in some way too, not just ω_1.

Solution for Rotation about a Single Axis when All Reaction Wheels are Spinning

Take $\boldsymbol{\Omega} = \Omega_1 \boldsymbol{i}$, but allow for ω_1, ω_2 and ω_3 to all vary.

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + C_R\dot{\omega}_1 \\ 0 &= B\dot{\Omega}_2 + C_R\dot{\omega}_2 - C_R\Omega_1\omega_3 \\ 0 &= C\dot{\Omega}_3 + C_R\dot{\omega}_3 + C_R\Omega_1\omega_2 \end{aligned} \tag{26}$$

Here, to keep Ω_2 and Ω_3 at zero, this requires:

$$\begin{aligned} C_R\dot{\omega}_2 - C_R\Omega_1\omega_3 &= 0 \\ C_R\dot{\omega}_3 + C_R\Omega_1\omega_2 &= 0 \end{aligned} \tag{27}$$

Again, this is a pair of coupled first order equations giving similar solutions to previous equations, except that now the frequency depends on Ω_1 which may vary in time.

If Ω_1 is constant, then the solution is:

$$\begin{aligned} \omega_2 &= R\sin(\Omega_1 t + \phi) \\ \omega_3 &= R\cos(\Omega_1 t + \phi) \end{aligned} \tag{28}$$

Where, in order to satisfy initial conditions:

$$R = \sqrt{\omega_{2,0}^2 + \omega_{3,0}^2} \qquad \phi = \arctan\left(\frac{\omega_{2,0}}{\omega_{3,0}}\right)$$

If Ω_1 is allowed to change, then the solution is:

$$\begin{aligned} \omega_2 &= R\sin(\theta + \phi) \\ \omega_3 &= R\cos(\theta + \phi) \end{aligned} \tag{29}$$

Where θ is the integral of Ω_1 (the net rotation about $\boldsymbol{i}$):

$$\theta(t) = \int_{\tau=0}^{t} \Omega_1(\tau)d\tau \tag{30}$$

Therefore, this solution is valid even if Ω_1 varies in response to changes in ω_1.

Intuitive Understanding of this Solution

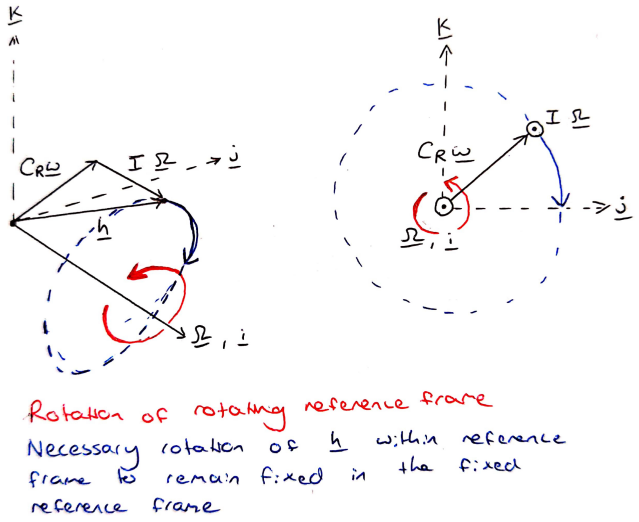

Figure 11: Vector diagram of the angular momentum within the rotating reference frame.

Figure 11 illustrates how the angular momentum must change within the rotating reference frame (aligned with the principle axis of the space station) in order for angular momentum to remain constant in the fixed reference frame. In order to do this, angular momentum must rotate about $\boldsymbol{i}$ in the opposite direction to Ω_1. The only quantities which contribute to components of $\boldsymbol{h}$ out of the $\boldsymbol{i}$ axis are ω_2 and ω_3.

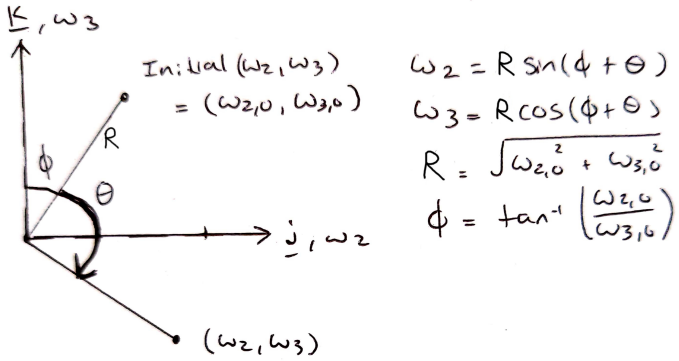

Figure 12: Demonstrates how ω_2 and ω_3 must rotate to counter the rotation from Ω_1.

If (ω_2, ω_3) must rotate by $\theta = \int \Omega_1 dt$ in the opposite direction of Ω_1, then by geometry, $\omega_2 = R\sin(\phi + \theta)$ and $\omega_3 = R\cos(\phi + \theta)$, where R and ϕ depend on the initial conditions as seen before.

Stability

Adding perturbations in Ω_2 and Ω_3 to the previous solution gives:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + C_R\dot{\omega}_1 + [C_R(\delta\Omega_2\omega_3 - \delta\Omega_3\omega_2)] \\ 0 &= C_R\dot{\omega}_2 - C_R\Omega_1\omega_3 + [B\dot{\Omega}_2 + (A-C)\Omega_1\dot{\Omega}_3 + C_R\delta\Omega_3\omega_1] \\ 0 &= C_R\dot{\omega}_3 - C_R\Omega_1\omega_2 + [C\dot{\Omega}_3 + (B-A)\Omega_1\dot{\Omega}_2 - C_R\delta\Omega_3\omega_2] \end{aligned} \tag{31}$$

The terms indicated with [ ] are the new contributions from the perturbations. The solution is identical to the system with a single reaction wheel, where stability requires ω^2 from eq. (25) to be positive.

The only difference is the extra term in the first line of eq. (31) which gives a contribution to $\dot{\Omega}_1$ of:

$$-\frac{C_R}{A}(\delta\Omega_2\omega_3 - \delta\Omega_3\omega_2)$$

Although this term isn't zero, it doesn't change Ω_1 significantly, so can be neglected. However, a more accurate control system could measure Ω_2 and Ω_3 and account for these in the value of $\dot{\omega}_1$ in order to counteract the perturbations.

5.4 Motion of the Spacecraft with Control Moment Gyroscopes

5.4.1 Behavoiur of a Standard Gyroscope

To give an idea of the effect of the gyroscopes on the spacecraft, it is useful to investigate the behaviour of a standard gyroscope.

The gyroscope looked at is identical to the one in the lab, mounted along an axis through the centre of mass. Gimbal inertia will be ignored, and the rotor has the same moments of inertia as the previous rotors looked at.

The reference frame used will be aligned with the spin axis, with angular velocity $\boldsymbol{\Omega}$. The rotor is spinning with a rate of ω about the $\boldsymbol{k}$ axis. Therefore, the angular momentum is given by:

$$\boldsymbol{h} = I_R(\boldsymbol{\Omega} + \omega\boldsymbol{k}) \tag{32}$$

Differentiating this in a rotating reference frame, and substituting into $\boldsymbol{Q} = \dot{\boldsymbol{h}}$ gives:

$$\boldsymbol{Q} = I_R\dot{\boldsymbol{\Omega}} + C_R\dot{\omega}\boldsymbol{k} + \Omega \times I_R\Omega + \Omega \times C_R\omega\boldsymbol{k} \tag{33}$$

Which when expanded out, gives:

$$\begin{aligned} Q_1 &= A_R\dot{\Omega}_1 + [(C_R - A_R)\Omega_3 + C_R\omega]\Omega_2 \\ Q_2 &= A_R\dot{\Omega}_2 - [(C_R - A_R)\Omega_3 + C_R\omega]\Omega_1 \\ Q_3 &= C_R(\dot{\Omega}_3 + \dot{\omega}) \end{aligned} \tag{34}$$

Since the gyroscopes will be operating with large rotor spin, $C_R\omega >> (C_R - A_R)\Omega_3$, this can be simplified to:

$$\begin{aligned} Q_1 &= A_R\dot{\Omega}_1 + C_R\omega\Omega_2 \\ Q_2 &= A_R\dot{\Omega}_2 - C_R\omega\Omega_1 \\ Q_3 &= C_R(\dot{\Omega}_3 + \dot{\omega}) \end{aligned} \tag{35}$$

Using Euler Angles

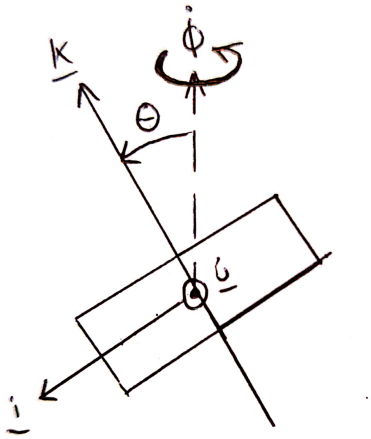

Figure 13: Description of gyroscope spin axis using Euler angles.

If the spin axis is described in terms of euler angles as shown in Figure 13, then $\boldsymbol{\Omega}$ is equal to:

$$\boldsymbol{\omega} = \begin{pmatrix} -\dot{\phi}\sin\theta \\ \dot{\omega} \\ \dot{\phi}\cos\theta \end{pmatrix} \tag{36}$$

Substituting this into eq. (35) gives:

$$\begin{aligned} Q_1 &= -A_R(\ddot{\phi} + \dot{\phi}\dot{\theta}\cos\theta) + C_R\omega\dot{\theta} \\ Q_2 &= A_R\dot{\theta} + C_R\omega\dot{\phi}\sin\theta \\ Q_3 &= C_R(\ddot{\phi}\cos\theta - \dot{\phi}\dot{\theta}\sin\theta + \dot{\omega}) \end{aligned} \tag{37}$$

Equilibrium Solution with an Applied Couple

If no couple is applied to the system, then it will remain stationary. However, if a couple Q_2 is applied then line 2 of eq. (37) gives:

$$Q_2 = C_R\omega\dot{\phi}\sin\theta \tag{38}$$

Therefore, the precession rate is constant and depends on the applied couple, rotor spin and spin axis inclination:

$$\dot{\phi} = \frac{Q_2}{C_R\omega\sin\theta} \tag{39}$$

By attaching these gyroscopes to a spacecraft, the phenomenon of precession can be utilised to rotate the spacecraft.

5.4.2 Structure of the Control Moment Gyroscope System

The structure of the control moment gyroscope system is similar to the reaction wheel system, except that instead of varying the spin of the rotors, the spin axes are tilted instead.

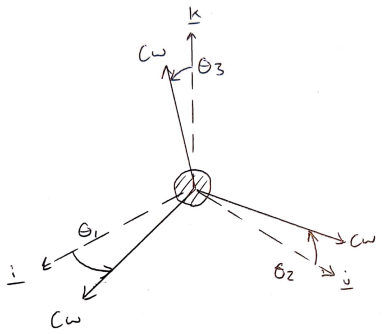

Figure 14: Description of control moment gyroscope system, with inclination of spin axes labelled.

The rotors have identical, constant spin ω. The gyroscopes are mounted in single-gimbals such that each rotor axis is able to tilt in a single direction. The inclinations are θ_1, θ_2 and θ_3, each about a different axis.

5.4.3 Equations of Motion

The total moment of inertia can be given by eq. (40) below:

$$\boldsymbol{h} = I\boldsymbol{\Omega} + C_R\omega \begin{pmatrix} \cos\theta_1 + \sin\theta_3 \\ \cos\theta_2 + \sin\theta_1 \\ \cos\theta_3 + \sin\theta_2 \end{pmatrix} \tag{40}$$

In deriving this, it has been assumed that the angular momentum of the rotors can neglect the contribution from the rotation $\boldsymbol{\Omega}$ of the spacecraft. This is sensible, since the spacecraft will have much larger moments of inertia than the rotor. Therefore the angular momentum of the rotors $C_R\omega$ can simply be mapped onto the principle axes of the spacecraft.

Differentiating this in a rotating reference frame using eq. (4) and substituting into eq. (3) ($\boldsymbol{Q} = \dot{h} = 0$) will give the equations of motions. To simplify this equation to something more useful, it can be assumed that the inclination angles are small and second order terms can be ignored, giving:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + (C-B)\Omega_2\Omega_3 + C_R\omega\dot{\theta}_3 + C_R\omega(\Omega_2 - \Omega_3) \\ 0 &= B\dot{\Omega}_2 + (A-C)\Omega_1\Omega_3 + C_R\omega\dot{\theta}_1 + C_R\omega(\Omega_3 - \Omega_1) \\ 0 &= C\dot{\Omega}_3 + (B-A)\Omega_1\Omega_2 + C_R\omega\dot{\theta}_2 + C_R\omega(\Omega_1 - \Omega_2) \end{aligned} \tag{41}$$

5.4.4 Controlling Rotation

To get an idea of the behaviour of the system, the initial behaviour when starting from rest can be considered, where $\boldsymbol{\Omega} = \boldsymbol{0}$. This gives:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + C_R\omega\dot{\theta}_3 \\ 0 &= B\dot{\Omega}_2 + C_R\omega\dot{\theta}_1 \\ 0 &= C\dot{\Omega}_3 + C_R\omega\dot{\theta}_2 \end{aligned} \tag{42}$$

Therefore, by changing the inclination of one of the axis, this will change the angular velocity of the body about another axis. For example, Ω_1 is controlled by θ_3 according to:

$$\begin{aligned} \dot{\Omega}_1 &= -\frac{C_R}{A}\dot{\theta}_3 \\ \Delta\Omega_1 &= -\frac{C_R}{A}\Delta\theta_3 + \text{Constant} \end{aligned} \tag{43}$$

However, once Ω_1 becomes non-zero, this changes the solution:

$$\begin{aligned} 0 &= A\dot{\Omega}_1 + C_R\omega\dot{\theta}_3 \\ 0 &= B\dot{\Omega}_2 + C_R\omega\dot{\theta}_1 - C_R\omega\Omega_1 \\ 0 &= C\dot{\Omega}_3 + C_R\omega\dot{\theta}_2 + C_R\omega\Omega_1 \end{aligned} \tag{44}$$

Once Ω_1 is non-zero, this causes Ω_2 and Ω_3 to change. From eq. (43) and eq. (44), assuming that both Ω_1 and θ_1 were initally zero, this gives:

$$\begin{aligned} \dot{\Omega}_2 &= \frac{C_R\omega}{B}\Omega_1 = -\frac{C_R^2\omega}{AB}\theta_3 \\ \dot{\Omega}_3 &= -\frac{C_R\omega}{C}\Omega_1 = \frac{C_R^2\omega}{AC}\theta_3 \end{aligned} \tag{45}$$

If $AB >> C_R^2\omega$ and $AC >> C_R^2\omega$ then these angular velocities can be made small, but will not be zero. Therefore, this method of controlling the spacecraft motion can only be active for short periods of time while Ω_2 and Ω_3. If these angular velocities become large, the motion will become unpredictable. Alternatively, more complex control methods could be designed to counter these off-axis angular velocities.

Figure 8 and Figure 9 show a controlled and uncontrolled change in attitude by attempting to rotate about a single axis.

5.4.5 Comparison with a Standard Gyroscope

The control moment gyroscopes utilise precession to rotate the spacecraft body, in the same way that a gyroscope will undergo precession if an external couple is applied. When the spin axis of a rotor is tilted, in order to keep the net angular momentum constant, the whole system must increase its angular momentum in the opposite direction to this, which is done by rotating the system using precessoin

In order to tilt the axis, a couple must be applied to the CMG. When considering the whole system together, this is an internal couple so doesn't appear in the equations of motion.

However, if the spacecraft and CMG were viewed separately, this would show that when the spacecraft applies a couple to rotate the spin axis, an equal and opposite couple is exerted on the spacecraft, accelerating it and causing it to start precessing.

6 Conclusions

1. For an ABC body, motion about a principle axis is stable if the corresponding moment of inertia is a minimum or maximum, but unstable if it is the intermediate one.

2. Both reaction wheels and control moment gyroscopes use conservation of angular momentum to manipulate the angular velocity of the spacecraft. Reaction wheels do this by changing the spin of rotors with fixed spin axes, while control moment gyroscopes do this by tilting the spin axis of rotors, which have constant spin.

3. If using a single reaction wheel, the attitude control method is simple. Any change in the angular momentum of the wheel will produce an equal and opposite change in the angular momentum of the spacecraft. This solution is stable for most situations, becoming unstable when rotating about the spacecraft intermediate axis with a large initial angular momentum about this axis.

4. If using multiple reaction wheels, the attitude control method needs to be modified to vary the spin of the other two reaction wheels to counteract the rotation of the spacecraft and keep angular momentum constant.

5. For control moment gyroscopes, tilting the spin axis will produce rotation along an axis perpendicular to the spin axis, by precession. This method requiers that precession occurs for a short period of time. If not, angular velocity is produced in the other two axes, causing the motino to become uncontrolled.

7 References

[1] Fraser Cain, 13/08/19, *Spacecraft Gyroscopes and Reaction Wheels. You Can Never Have Enough* https://www.universetoday.com/143152/spacecraft-gyroscopes-and-reaction-wheels-you-can-never-have-enough/ (Accessed: 05/12/19)

[2] Nasa, *Hubble Space Telescope - Pointing Control System* https://www.nasa.gov/content/goddard/hubble-space-telescope-pointing-control-system (Accessed: 05/12/19)

[3] Karthikeyan KC, 03/08/16, *How Reaction Wheels and Control Moment Gyros Work?* https://geekswipe.net/technology/aerospace/how-reaction-wheels-and-control-moment-gyros-work/ (Accessed: 05/12/19)

[4] Erik Cheever, *Fourth Order Runge-Kutta* https://lpsa.swarthmore.edu/NumInt/NumIntFourth.html (Accessed: 06/12/19)

[5] Meriam J.L., Kraige L.G., 2012, *Engineering Mechanics: Dynamics, 7th edition*, Chapter 7, pp.541

[6] Meriam J.L., Kraige L.G., 2012, *Engineering Mechanics: Dynamics, 7th edition*, Chapter 4, pp.274, equation (4/11)

[7] Meriam J.L., Kraige L.G., 2012, *Engineering Mechanics: Dynamics, 7th edition*, Chapter 7, pp.527, equation (7/4)

[8] Meriam J.L., Kraige L.G., 2012, *Engineering Mechanics: Dynamics, 7th edition*, Appendix B, pp.643, equation (B/3)

8 Appendix

8.1 Code for Integrating the Equations of Motion for The Spacecraft with No Actuation Devices

```
"""
This script integrates the equations of motion for the spacecraft with no actuation devices,
as a simple ABC body.
"""

import numpy as np
import matplotlib.pyplot as plt


""" Classes and Functions for Integration and Plotting """

class Properties:

    def __init__(self, A=0, B=0, C=0):
        self.A = A
        self.B = B
        self.C = C

    def unpack(self):
        return self.A, self.B, self.C

def simulate(properties, initial_omega, T, N):

    A, B, C = properties.unpack()

    def f(t, omega):
        omega_dot_1 = omega[1]*omega[2] * (C-B)/A
        omega_dot_2 = omega[0]*omega[2] * (A-C)/B
        omega_dot_3 = omega[0]*omega[1] * (B-A)/C
        return np.array([omega_dot_1, omega_dot_2, omega_dot_3])

    omega = np.zeros((N, 3))
    omega[0] = initial_omega

    t = np.linspace(0, T, N)
    dt = T/(N-1)

    for n in range(N-1):
        k1 = f(t[n], omega[n])
        k2 = f(t[n] + dt/2, omega[n] + k1*dt/2)
        k3 = f(t[n] + dt/2, omega[n] + k2*dt/2)
        k4 = f(t[n] + dt, omega[n] + k3*dt)
        omega[n+1] = omega[n] + (1/6)*(k1 + 2*k2 + 2*k3 + k4)*dt

    return t, omega

def plot_simulation(t, omega, name=None):

    plt.figure(figsize=(7, 5))

    plt.plot(t, omega[:, 0], label=r'$\Omega_1$')
    plt.plot(t, omega[:, 1], label=r'$\Omega_2$')
    plt.plot(t, omega[:, 2], label=r'$\Omega_3$')
    plt.legend(fontsize=14)
```

```
    plt.xlabel('Time t (s)', fontsize=14)
    plt.ylabel(r'$\mathbf{\Omega}$ (rad/s)', fontsize=14)

    if name is not None:
        plt.savefig('../fig/{}'.format(name), bbox_inches='tight')
    plt.show()


""" Test Cases """

# Stable motion of an ABC with constant angular velocity about a principle axis.

properties = Properties(100, 150, 200)
initial_omega = np.array([10, 0.1, -0.2])
T = 10
N = 100
t, omega = simulate(properties, initial_omega, T, N)
plot_simulation(t, omega, 'no_actuation_ABC_stable.png')


# Unstable motion of an ABC body.

properties = Properties(100, 150, 200)
initial_omega = np.array([0.1, 10, -0.2])
T = 10
N = 100
t, omega = simulate(properties, initial_omega, T, N)
plot_simulation(t, omega, 'no_actuation_ABC_unstable.png')
```

8.2 Code for Integrating the Equations of Motion for The Spacecraft with Reaction Wheels

```
"""
This script integrates the equations of motion for the reaction wheels system, for a series
of situations.
"""

import numpy as np
import matplotlib.pyplot as plt


""" Classes and Functions for Integration and Plotting """

class Properties:

    def __init__(self, A=0, B=0, C=0, CR=0):
        self.A = A
        self.B = B
        self.C = C
        self.CR = CR

    def unpack(self):
        return self.A, self.B, self.C, self.CR


class Conditions:

    def __init__(self, omega_initial=None,
                 get_spin = lambda t: np.zeros(3),
```

```
                get_spin_acc = lambda t: np.zeros(3)):
        self.omega_initial = omega_initial
        self.get_spin = get_spin
        self.get_spin_acc = get_spin_acc

    def unpack(self):
        return self.omega_initial, self.get_spin, self.get_spin_acc


def simulate(properties, conditions, T, N):

    A, B, C, CR = properties.unpack()
    omega_initial, get_spin, get_spin_acc = conditions.unpack()

    def f(t, omega):

        spin = get_spin(t)
        spin_acc = get_spin_acc(t)

        omega_1_acc = -(1/A)*(CR*spin_acc[0] + (C-B)*omega[1]*omega[2]
                            + CR*(omega[1]*spin[2] - omega[2]*spin[1]))
        omega_2_acc = -(1/B)*(CR*spin_acc[1] + (A-C)*omega[0]*omega[2]
                            + CR*(omega[2]*spin[0] - omega[0]*spin[2]))
        omega_3_acc = -(1/C)*(CR*spin_acc[2] + (B-A)*omega[0]*omega[1]
                            + CR*(omega[0]*spin[1] - omega[1]*spin[0]))

        return np.array([omega_1_acc, omega_2_acc, omega_3_acc])

    omega = np.zeros((N, 3))
    omega[0] = omega_initial

    t = np.linspace(0, T, N)
    dt = T/(N-1)

    for n in range(N-1):
        k1 = f(t[n], omega[n])
        k2 = f(t[n] + dt/2, omega[n] + k1*dt/2)
        k3 = f(t[n] + dt/2, omega[n] + k2*dt/2)
        k4 = f(t[n] + dt, omega[n] + k3*dt)

        omega[n+1] = omega[n] + (1/6)*(k1 + 2*k2 + 2*k3 + k4)*dt

    return t, omega, \
           np.array([get_spin(ti) for ti in t]), \
           np.array([get_spin_acc(ti) for ti in t])

def plot_simulation(t, omega, spin, name=None):

    fig, (ax0, ax1) = plt.subplots(1, 2, figsize=(9, 5))
    fig.tight_layout()
    fig.subplots_adjust(wspace=0.25)

    ax0.plot(t, spin[:, 0], label=r'$\omega_1$')
    ax0.plot(t, spin[:, 1], label=r'$\omega_2$')
    ax0.plot(t, spin[:, 2], label=r'$\omega_3$')

    ax0.legend(fontsize=14)
    ax0.set_xlabel('Time t (s)', fontsize=14)
    ax0.set_ylabel(r'$\mathbf{\omega}$ (rad/s)', fontsize=14)
```

```
    ax1.plot(t, omega[:, 0], label=r'$\Omega_1$')
    ax1.plot(t, omega[:, 1], label=r'$\Omega_2$')
    ax1.plot(t, omega[:, 2], label=r'$\Omega_3$')
    ax1.legend(fontsize=14)
    ax1.set_xlabel('Time t (s)', fontsize=14)
    ax1.set_ylabel(r'$\mathbf{\Omega}$ (rad/s)', fontsize=14)

    if name is not None:
        plt.savefig('../fig/{}'.format(name), bbox_inches='tight')
    plt.show()


""" Builder functions for giving pairs of spin and spin acceleration functions """


def build_constant_spin(spin1, spin2, spin3):
    get_spin = lambda t: np.array([spin1, spin2, spin3])
    get_spin_acc = lambda t: np.array([0, 0, 0])
    return get_spin, get_spin_acc

def build_spin_ramp(index, change, start, length):
    def get_spin(t):
        if t<start:
            spin = 0
        elif t<start+length:
            spin = change*(t-start)/length
        else:
            spin = change
        spins = np.zeros(3)
        spins[index] = spin
        return spins

    def get_spin_acc(t):
        if t>start and t<start+length:
            spin_acc = change/length
        else:
            spin_acc = 0
        spin_accs = np.zeros(3)
        spin_accs[index] = spin_acc
        return spin_accs

    return get_spin, get_spin_acc

def combine_pairs(*pairs):
    def get_spin(t):
        total = 0
        for pair in pairs:
            total += pair[0](t).astype(float)
        return total

    def get_spin_acc(t):
        total = 0
        for pair in pairs:
            total += pair[1](t).astype(float)
        return total

    return get_spin, get_spin_acc


""" Test Cases """
```

```
# Stable rotation about a single axis, using a single reaction wheel

properties = Properties(100, 150, 200, 0.5)
T = 10
N = 1000
omega_initial = np.array([0, 0.02, -0.01])

def get_spin(t):
    if t<1 or t>9:
        return np.zeros(3)
    else:
        spin1 = 20*(np.cos(2*np.pi*(t-1)/8)-1)
        return np.array([spin1, 0, 0])

def get_spin_acc(t):
    if t<1 or t>9:
        return np.zeros(3)
    else:
        spin1_acc = -20*(2*np.pi/5)*np.sin(2*np.pi*(t-1)/8)
        return np.array([spin1_acc, 0, 0])

conditions = Conditions(omega_initial, get_spin, get_spin_acc)
t, omega, spin, spin_acc = simulate(properties, conditions, T, N)
plot_simulation(t, omega, spin, 'reaction_single_zero.png')


# Unstable motion when attempting to rotate using a single reaction wheel

properties = Properties(100, 50, 300, 1)
T = 30
N = 1000
omega_initial = np.array([0, 0.005, -0.002])

get_spin, get_spin_acc = combine_pairs(
    build_constant_spin(50, 0, 0),
    build_spin_ramp(0, -60, 1, 2)
)

conditions = Conditions(omega_initial, get_spin, get_spin_acc)
t, omega, spin, spin_acc = simulate(properties, conditions, T, N)
plot_simulation(t, omega, spin, 'reaction_single_zero_unstable.png')


# Motion when attempting to re-use previous method of rotation, when other
# reaction wheels are spinning too

properties = Properties(100, 150, 200, 0.5)
T = 10
N = 1000
omega_initial = np.array([0, 0, 0])

def get_spin(t):
    if t<1 or t>9:
        return np.zeros(3)
    else:
        spin1 = 1.5*(np.cos(2*np.pi*(t-1)/8)-1)
        return np.array([spin1, 0, 0])

def get_spin_acc(t):
```

```
    if t<1 or t>9:
        return np.zeros(3)
    else:
        spin1_acc = -1.5*(2*np.pi/5)*np.sin(2*np.pi*(t-1)/8)
        return np.array([spin1_acc, 0, 0])

get_spin, get_spin_acc = combine_pairs(
    build_constant_spin(20, 18, 19),
    (get_spin, get_spin_acc)
)
conditions = Conditions(omega_initial, get_spin, get_spin_acc)
t, omega, spin, spin_acc = simulate(properties, conditions, T, N)
plot_simulation(t, omega, spin, 'reaction_naive_attempt.png')


# Solution for rotating about a single axis with constant angular velocity
# when all reaction wheels are spinning

properties = Properties(100, 150, 200, 0.5)

T = 10
N = 100

omega1 = 1
omega2 = 0
omega3 = 0
omega_initial = np.array([omega1, omega2, omega3])
initial_spin = np.array([10.0, -20.0, 30.0])

def get_spin(t):
    spin1 = initial_spin[0]
    R = np.sqrt(initial_spin[1]**2 + initial_spin[2]**2)
    phi = np.arctan2(initial_spin[2], initial_spin[1])
    spin2 = R*np.sin(omega1*t + phi)
    spin3 = R*np.cos(omega1*t + phi)
    return np.array([spin1, spin2, spin3])

def get_spin_acc(t):
    spin1 = 0
    R = np.sqrt(initial_spin[1]**2 + initial_spin[2]**2)
    phi = np.arctan2(initial_spin[2], initial_spin[1])
    spin2 = omega1*R*np.cos(omega1*t + phi)
    spin3 = -omega1*R*np.sin(omega1*t + phi)
    return np.array([spin1, spin2, spin3])


conditions = Conditions(omega_initial, get_spin, get_spin_acc)
t, omega, spin, spin_acc = simulate(properties, conditions, T, N)
plot_simulation(t, omega, spin, 'reaction_smart_constant.png')


# Solution for rotating about a single axis with variable angular velocity
# when all reaction wheels are spinning

properties = Properties(100, 150, 200, 0.5)
A, B, C, CR = properties.unpack()

T = 20
N = 100
```

```
omega_initial = np.array([0, 0, 0])
initial_spin = np.array([0, -4, 5])

def get_delta_spin1_derivative(t):
    if t<1 or t>19:
        return 0
    else:
        return -(2*np.pi/18)*40*np.sin(2*np.pi*(t-1)/18)

def get_delta_spin1(t):
    if t<1 or t>19:
        return 0
    else:
        return 40*(np.cos(2*np.pi*(t-1)/18)-1)

def get_delta_spin1_integral(t):
    if t<1:
        return 0
    elif t<19:
        return (18/(2*np.pi))*40*(np.sin(2*np.pi*(t-1)/18)-t)
    else:
        return (18/(2*np.pi))*40*(np.sin(2*np.pi*(18-1)/18)-18)

def get_omega1(t):
    return - (CR/A)*get_delta_spin1(t) + omega_initial[0]

def get_omega1_integral(t):
    return - (CR/A)*get_delta_spin1_integral(t) + omega_initial[0]*t

def get_spin(t):
    spin1 = get_delta_spin1(t) + initial_spin[0]
    R = np.sqrt(initial_spin[1]**2 + initial_spin[2]**2)
    phi = np.arctan2(initial_spin[2], initial_spin[1])
    theta = get_omega1_integral(t)
    spin2 = R*np.sin(theta + phi)
    spin3 = R*np.cos(theta + phi)
    return np.array([spin1, spin2, spin3])

def get_spin_acc(t):
    spin1 = get_delta_spin1_derivative(t)
    R = np.sqrt(initial_spin[1]**2 + initial_spin[2]**2)
    phi = np.arctan2(initial_spin[2], initial_spin[1])
    omega1 = get_omega1(t)
    theta = get_omega1_integral(t)
    spin2 = omega1*R*np.cos(theta + phi)
    spin3 = -omega1*R*np.sin(theta + phi)
    return np.array([spin1, spin2, spin3])


conditions = Conditions(omega_initial, get_spin, get_spin_acc)
t, omega, spin, spin_acc = simulate(properties, conditions, T, N)
plot_simulation(t, omega, spin, 'reaction_smart_variable.png')
```

8.3 Code for Integrating the Equations of Motion for The Spacecraft with Control Moment Gyroscopes

```
"""
This script integrates the equations of motion for the control moment gyroscope system, for
a series of situations.
"""

import numpy as np
import matplotlib.pyplot as plt


""" Classes and Functions for Integration and Plotting """

class Properties:

    def __init__(self, A, B, C, CR):
        self.A = A
        self.B = B
        self.C = C
        self.CR = CR

    def unpack(self):
        return self.A, self.B, self.C, self.CR

class Conditions:

    def __init__(self, omega_initial=None,
                 spin = 10,
                 get_theta = lambda t: np.zeros(3),
                 get_theta_acc = lambda t: np.zeros(3)):
        self.omega_initial = omega_initial
        self.spin = spin
        self.get_theta = get_theta
        self.get_theta_acc = get_theta_acc

    def unpack(self):
        return self.omega_initial, self.spin, self.get_theta, self.get_theta_acc


def simulate(properties, conditions, T, N):

    A, B, C, CR = properties.unpack()
    omega_initial, spin, get_theta, get_theat_acc = conditions.unpack()

    def f(t, omega):

        theta = get_theta(t)
        theta_acc = get_theta_acc(t)

        omega_1_acc = -(1/A)*((C-B)*omega[1]*omega[2]
                              + CR*spin*theta_acc[2]
                              + CR*spin*(omega[1]*(np.cos(theta[2]) + np.sin(theta[1]))
                                         -omega[2]*(np.cos(theta[1]) + np.sin(theta[0]))))
        omega_2_acc = -(1/B)*((A-C)*omega[0]*omega[2]
                              + CR*spin*theta_acc[0]
                              + CR*spin*(omega[2]*(np.cos(theta[0]) + np.sin(theta[2]))
                                         -omega[0]*(np.cos(theta[2]) + np.sin(theta[1]))))
        omega_3_acc = -(1/C)*((B-A)*omega[0]*omega[1]
```

```
                                + CR*spin*theta_acc[1]
                                + CR*spin*(omega[0]*(np.cos(theta[1]) + np.sin(theta[0]))
                                           -omega[1]*(np.cos(theta[0]) + np.sin(theta[2])))

        return np.array([omega_1_acc, omega_2_acc, omega_3_acc])

    omega = np.zeros((N, 3))
    omega[0] = omega_initial

    t = np.linspace(0, T, N)
    dt = T/(N-1)

    for n in range(N-1):
        k1 = f(t[n], omega[n])
        k2 = f(t[n] + dt/2, omega[n] + k1*dt/2)
        k3 = f(t[n] + dt/2, omega[n] + k2*dt/2)
        k4 = f(t[n] + dt, omega[n] + k3*dt)

        omega[n+1] = omega[n] + (1/6)*(k1 + 2*k2 + 2*k3 + k4)*dt

    return t, omega, \
           np.array([get_theta(ti) for ti in t])

def plot_simulation(t, omega, theta, name=None):

    fig, (ax0, ax1) = plt.subplots(1, 2, figsize=(9, 5))
    fig.tight_layout()
    fig.subplots_adjust(wspace=0.25)

    ax0.plot(t, theta[:, 0], label=r'$\theta_1$')
    ax0.plot(t, theta[:, 1], label=r'$\theta_2$')
    ax0.plot(t, theta[:, 2], label=r'$\theta_3$')

    ax0.legend(fontsize=14)
    ax0.set_xlabel('Time t (s)', fontsize=14)
    ax0.set_ylabel(r'$\mathbf{\theta}$ (rad)', fontsize=14)

    ax1.plot(t, omega[:, 0], label=r'$\Omega_1$')
    ax1.plot(t, omega[:, 1], label=r'$\Omega_2$')
    ax1.plot(t, omega[:, 2], label=r'$\Omega_3$')
    ax1.legend(fontsize=14)
    ax1.set_xlabel('Time t (s)', fontsize=14)
    ax1.set_ylabel(r'$\mathbf{\Omega}$ (rad/s)', fontsize=14)

    if name is not None:
        plt.savefig('../fig/{}'.format(name), bbox_inches='tight')
    plt.show()


""" Test Cases """

# Controlled rotation about a single axis

properties = Properties(100, 150, 200, 1)
T = 3
N = 100
omega_initial = np.array([0, 0, 0])
spin = 10

theta_T = 1
```

```
amp = 0.1

def get_theta(t):
    if t<theta_T:
        theta3 = amp*0.5*(1 - np.cos(2*np.pi*t/theta_T))
    else:
        theta3 = 0
    return np.array([0, 0, theta3])

def get_theta_acc(t):
    if t<theta_T:
        theta3_acc = (2*np.pi/theta_T)*amp*0.5*np.sin(2*np.pi*t/theta_T)
    else:
        theta3_acc = 0
    return np.array([0, 0, theta3_acc])

conditions = Conditions(omega_initial, spin, get_theta, get_theta_acc)
t, omega, theta = simulate(properties, conditions, T, N)
plot_simulation(t, omega, theta, 'cmg_controlled.png')


# Uncontrolled motion

properties = Properties(100, 150, 200, 1)
T = 40
N = 100
omega_initial = np.array([0, 0, 0])
spin = 10

theta_T = 20
amp = 1

def get_theta(t):
    if t<theta_T:
        theta3 = amp*0.5*(1 - np.cos(2*np.pi*t/theta_T))
    else:
        theta3 = 0
    return np.array([0, 0, theta3])

def get_theta_acc(t):
    if t<theta_T:
        theta3_acc = (2*np.pi/theta_T)*amp*0.5*np.sin(2*np.pi*t/theta_T)
    else:
        theta3_acc = 0
    return np.array([0, 0, theta3_acc])

conditions = Conditions(omega_initial, spin, get_theta, get_theta_acc)
t, omega, theta = simulate(properties, conditions, T, N)
plot_simulation(t, omega, theta, 'cmg_uncontrolled.png')
```